

L'extension

listofitems

v1.2a
18 novembre 2016

Christian TELLECHEA^{*}
Steven B. SEGLETES[†]

Cette petite extension pour est destinée à lire une liste d'éléments dont le séparateur peut être choisi par l'utilisateur. Une fois la liste lue, ses éléments sont stockés dans une structure qui se comporte comme un tableau unidimensionnel et ainsi, il devient très facile d'accéder à un élément de la liste par son numéro. Par exemple, si la liste est stockée dans la macro `\foo`, l'élément n° 3 est désigné par `\foo[3]`.

Un élément peut à son tour être une liste disposant d'un autre séparateur que celui de la liste de niveau supérieur, ce qui ouvre la voie à des imbrications et donne une syntaxe rappelant celle des tableaux à plusieurs dimensions du type `\foo[3,2]` pour accéder à l'élément n° 2 de la liste contenue dans l'élément n° 3 de la liste de plus haut niveau.

^{*}. unbonpetit@openmailbox.org

[†]. steven.b.segletes.civ@mail.mil

1 Avant-propos

Cette extension ne requiert aucun package, doit être utilisée avec un moteur ε -TeX, et doit être appelée sous (pdf)(Xe)(lua)TeX par

```
\usepackage{listofitems}
```

et sous (pdf)(Xe)(lua)TeX par

```
\input listofitems.tex
```

2 Lire une liste simple

Définir le séparateur Le $\langle \text{séparateur} \rangle$ par défaut est la virgule et si l'on souhaite en changer, il faut, avant de lire une liste d'éléments, le définir avec `\setsepchar{ $\langle \text{séparateur} \rangle$ }`. Un $\langle \text{séparateur} \rangle$ est un ensemble de tokens dont les catcodes sont différents de 1 et 2 (les accolades ouvrantes et fermantes), 14 (habituellement %) et 15. Le token de catcode 6 (habituellement #) n'est accepté que s'il est suivi d'un entier auquel cas l'ensemble désigne l'argument d'une macro ; en aucun cas ce token ne doit se trouver seul dans le $\langle \text{séparateur} \rangle$. Des commandes peuvent se trouver dans cet ensemble de tokens, y compris la primitive `\par`.

Le $\langle \text{séparateur} \rangle$ « / » est réservé par défaut pour les listes imbriquées (voir page 3). Il ne faut donc pas écrire « `\setsepchar{/}` » car `listofitems` comprendrait que l'on souhaite lire une liste imbriquée. Pour définir « / » comme $\langle \text{séparateur} \rangle$ d'une liste simple, il faut, à l'aide de l'argument optionnel, choisir un autre $\langle \text{séparateur} \rangle$ de listes imbriquées, par exemple « . » et écrire « `\setsepchar[.]{/}` ».

Il n'est pas possible de choisir « | » comme $\langle \text{séparateur} \rangle$ car il entrerait en conflit avec l'opérateur logique **OU** noté « || » (voir plus bas). On peut cependant contourner cette limitation, à ses risques et périls, en écrivant « `\setsepchar{{|}}` ».

Lire la liste Pour lire la liste d'éléments, la commande `\readlist{ $\langle \text{macroliste} \rangle$ { $\langle \text{liste} \rangle$ }}` doit être appelée. Ce faisant, la $\langle \text{liste} \rangle$ est lue et les éléments sont stockés dans une macro, notée $\langle \text{macroliste} \rangle$ qui dès lors, se comporte comme un tableau composé des éléments de la $\langle \text{liste} \rangle$. Un élément est un ensemble de tokens dont les accolades *doivent* être équilibrées. Les tokens de catcode 6 seuls, 14 et 15 ne sont pas autorisés dans les listes.

Par exemple, pour définir la $\langle \text{macroliste} \rangle$ nommée `\foo`, on peut écrire

```
\setsepchar{,}
\readlist\foo{12,abc,x y ,{\bfseries z},,TeX,,!}
```

Si la $\langle \text{liste} \rangle$ est contenue dans une macro, alors cette macro est développée par `listofitems`. On peut donc écrire `\readlist{ $\langle \text{macroliste} \rangle$ { $\langle \text{macro} \rangle$ }}` ce qui donnerait

```
\setsepchar{,}
\def\liste{12,abc,x y ,{\bfseries z},,TeX,,!}
\readlist\foo\liste
```

La macro `\greadlist` agit comme `\readlist` mais effectue des assignations *globales* et par conséquent, la $\langle \text{macroliste} \rangle$ est utilisable hors du groupe où a été exécutée `\greadlist`.

Accéder à un élément La macro `\foo` attend un argument numérique *obligatoire* entre crochets, que nous notons i et qui désigne le rang de l'élément auquel on souhaite accéder. Ainsi, `\foo[1]` est³ « 12 ». De la même façon, `\foo[4]` est « `{\bfseries z}` ».

Le nombre i peut également être négatif auquel cas le comptage se fait à partir de la fin de la liste : -1 représente le dernier rang, -2 l'avant-dernier, etc. Si le nombre d'éléments est n , alors l'argument $-n$ est le premier élément.

3. Il faut 2 développements à `\foo[i]` pour obtenir l'élément n° i .

D'une façon générale, si une $\langle liste \rangle$ a une longueur n , alors l'index i peut se trouver dans l'intervalle $[1; n]$ ou $[-n; -1]$ et dans le cas contraire, une erreur de compilation survient.

Si l'index est vide, alors $\backslash foo[]$ se développe en la $\langle liste \rangle$ entière.

La macro $\backslash foosep$ est créée. Elle s'utilise avec la syntaxe $\backslash foosep[\langle index \rangle]$ et permet d'accéder au séparateur qui suit l'élément de rang $\langle index \rangle$. Le dernier séparateur (celui qui suit le dernier élément) est défini par $listofitems$. Si l' $\langle index \rangle$ est vide, $\backslash foosep[]$ a un développement vide.

Choisir plusieurs séparateurs possibles Pour spécifier plusieurs séparateurs possibles, il faut utiliser l'opérateur OU noté « $||$ ». On peut par exemple utiliser cette fonctionnalité pour isoler les termes dans une somme algébrique :

$\backslash setsepchar\{+ - \}$	
$\backslash readlist\terme\{17-8+4-11\}$	1) 17 (séparateur = -)
1) $\backslash terme[1]$ (séparateur = $\backslash termesep[1]$) $\backslash par$	2) 8 (séparateur = +)
2) $\backslash terme[2]$ (séparateur = $\backslash termesep[2]$) $\backslash par$	3) 4 (séparateur = -)
3) $\backslash terme[3]$ (séparateur = $\backslash termesep[3]$) $\backslash par$	4) 11 (séparateur = +)
4) $\backslash terme[4]$ (séparateur = $\backslash termesep[4]$)	

Nombre d'éléments Si l'on écrit $\backslash readlist\langle macroliste \rangle\{\langle liste \rangle\}$ alors la macro $\langle macroliste \rangle$ len contient⁴ le nombre d'éléments de la $\langle liste \rangle$. Dans l'exemple avec $\backslash foo$, la macro $\backslash foolen$ contient 8.

Afficher tous les éléments À des fins de débogage, la macro $\backslash showitems\langle macroliste \rangle$ compose tous les éléments d'une liste tandis que que sa version étoilée affiche ces éléments « détokénisés »⁵.

$\backslash showitems\foo\par$	$\boxed{12} \boxed{abc} \boxed{x y} \boxed{z} \boxed{\TeX} \boxed{!}$
$\backslash showitems*\foo$	$\boxed{12} \boxed{abc} \boxed{x y} \boxed{\backslash bfseries z} \boxed{\backslash TeX} \boxed{!}$

La présentation de chaque élément est confiée à la macro $\backslash showitemsmacro$ dont le code est

```
\newcommand\showitemsmacro[1]{%
  \begingroup\fbboxsep=0.25pt \fbboxrule=0.5pt \fbbox{\strut#1}\endgroup
  \hskip0.25em\relax}
```

Il est donc possible — et souhaitable — de la redéfinir si l'on cherche un autre effet.

La macro $\backslash fbox$ et ses dimensions afférentes $\backslash fbboxsep$ et $\backslash fbboxrule$ sont définies par $listofitems$ lorsqu'on ne compile pas sous L^AT_EX de façon à obtenir le même résultat qu'avec L^AT_EX.

Suppression des espaces extrêmes Par défaut, $listofitems$ lit et prend en compte le (ou les) espaces se trouvant au début et à la fin d'un élément. Pour que ces espaces soient ignorés lors de la lecture de la $\langle liste \rangle$, il faut exécuter la version étoilée $\backslash readlist*\langle macro \rangle\{\langle liste \rangle\}$:

$\backslash setsepchar\{, \}$	
$\backslash readlist*\foo\{12,abc, x y ,\backslash bfseries z\, ,\TeX,,!\}$	$\boxed{12} \boxed{abc} \boxed{x y} \boxed{z} \boxed{\TeX} \boxed{!}$
$\backslash showitems\foo$	

Gestion des éléments vides Par défaut, $listofitems$ prend en compte les éléments vides. Ainsi, dans l'exemple précédent, le 2-développement de $\backslash foo[7]$ est vide. Pour que des éléments vides (ceux délimités par deux séparateurs consécutifs dans la liste) soient ignorés, il faut, avant de lancer la macro $\backslash readlist$, exécuter la macro $\backslash ignoreemptyitems$. La macro $\backslash reademptyitems$ revient au comportement par défaut.

4. C'est-à-dire qu'elle est purement développable et se développe en un nombre

5. La primitive $\backslash detokenize$ qui procède à cette dénaturation insère un espace après chaque séquence de contrôle.

Cette option peut être utilisée seule ou combinée avec `\readlist*` auquel cas la suppression des espaces extrêmes intervient *avant* que `listofitems` n'ignore les éléments vides :

<pre>\setsepchar{,} \ignoreemptyitems \readlist\foo{12,abc, x y ,{\bfseries z}, ,\TeX,,!} a) nombre d'éléments = \foolen\par \showitems\foo \readlist*\foo{12,abc, x y ,{\bfseries z}, ,\TeX,,!} b) nombre d'éléments = \foolen\par \showitems\foo</pre>	<p>a) nombre d'éléments = 7</p> <pre>12 abc x y z \TeX !</pre> <p>b) nombre d'éléments = 6</p> <pre>12 abc x y z \TeX !</pre>
---	---

Itérer sur la liste Une fois une liste lue par `\readlist` et stockée dans une *macroliste*, la commande `\foreachitem` *variable* *in* *macroliste*{*code*} itère sur la liste : la *variable* est une macro choisie par l'utilisateur qui prendra tour à tour la valeur de chaque élément. La macro *cnt* représente le numéro de l'élément contenu dans *variable*.

<pre>\setsepchar{ }% séparateur = espace \readlist\phrase{Une phrase de test.} \foreachitem\mot\in\phrase{Le mot numéro \motcnt{} : \mot\par}</pre>	<p>Le mot numéro 1 : Une</p> <p>Le mot numéro 2 : phrase</p> <p>Le mot numéro 3 : de</p> <p>Le mot numéro 4 : test.</p>
---	---

Assigner un élément à une macro La commande `\itemtomacro`*macroliste*[*index*]*macro* assigne à la *macro* l'élément désigné par *macroliste*[*index*]. La *macro* ainsi définie est purement développable, sous réserve que l'élément qu'elle contient le soit.

<pre>\setsepchar{ }% séparateur = espace \readlist\phrase{Une phrase de test.} \itemtomacro\phrase[2]\unmot \meaning\unmot\par \itemtomacro\phrase[-1]\motdelafin \meaning\motdelafin</pre>	<p>macro:->phrase</p> <p>macro:->test.</p>
---	--

3 Listes imbriquées

On parle de liste « imbriquée » lorsque l'on demande à `listofitems` de lire une liste où les éléments sont à leur tour compris comme une liste (dont le séparateur est différent de la liste de niveau supérieur). Le nombre d'imbrication n'est pas limité, mais dans la pratique, un niveau d'imbrication de 2, voire 3, semble un maximum.

Définir les séparateurs Pour indiquer que les éléments de la liste doivent eux-mêmes être compris comme des listes et que la recherche des éléments sera récursive, il faut spécifier plusieurs *séparateurs*, chacun correspondant à un niveau d'imbrication. Pour déclarer une *liste de séparateurs* il faut définir le *séparateur* de cette *liste de séparateurs* à l'aide de l'argument optionnel de la macro `\setsepchar` et écrire `\setsepchar[séparateur]{liste des séparateurs}`.

Par défaut, le *séparateur* est « / ». Ainsi, si l'on donne l'ordre

```
\setsepchar{\\/,/ }
```

on indique une profondeur récursive de 3 et on choisit comme séparateur de la *liste des séparateurs* le caractère par défaut « / » :

- les éléments de niveau 1 sont trouvés entre les séparateurs « \\ » ;
- les éléments de niveau 2 sont trouvés dans les éléments de niveau 1 entre les séparateurs « , » ;
- enfin, les éléments de niveau 3 sont trouvés dans ceux de niveau 2 entre les séparateurs « _ ».

La *profondeur* de recherche est contenue dans la macro purement développable `\nestdepth`.

Lire et accéder aux éléments Pour les listes imbriquées, les index obéissent à la règle suivante :

- `[]` désigne la liste principale, c'est-à-dire l'argument de `\readlist` ;
- `[<i>]` désigne l'élément n° *i* de la liste principale ;

- [$\langle i \rangle$, $\langle j \rangle$] désigne l'élément n° $\langle j \rangle$ de la liste constituée par l'élément évoqué au point précédent;
- [$\langle i \rangle$, $\langle j \rangle$, $\langle k \rangle$] désigne l'élément n° $\langle k \rangle$ de la liste constituée par l'élément évoqué au point précédent;
- etc.

Comme pour les liste non imbriquées, les index peuvent être négatifs.

Pour lire les éléments, la syntaxe de `\readlist` est exactement la même qu'avec les listes simples :

<code>\setsepchar{\,/ }</code>	
<code>\readlist\baz{1,2 a b,3 c\4 d e f,5,6\7,,8, ,9 xy z}</code>	a) <code>\baz[1]</code> est 1,2 a b,3 c
a) <code>\string\baz[1]</code> est <code>\baz[1]\par</code>	b) <code>\baz[1,1]</code> est 1
b) <code>\string\baz[1,1]</code> est <code>\baz[1,1]\par</code>	c) <code>\baz[1,1,1]</code> est 1
c) <code>\string\baz[1,1,1]</code> est <code>\baz[1,1,1]\par</code>	b) <code>\bar[1,2]</code> est 2 a b
b) <code>\string\bar[1,2]</code> est <code>\baz[1,2]\par</code>	e) <code>\baz[1,2,3]</code> est b
e) <code>\string\baz[1,2,3]</code> est <code>\baz[1,2,3]\par</code>	f) <code>\baz[-2,1,-1]</code> est f
f) <code>\string\baz[-2,1,-1]</code> est <code>\baz[-2,1,-1]</code>	

L'opérateur « || » Cet opérateur peut se trouver dans n'importe quel niveau d'imbrication.

<code>\setsepchar[,]{+ -,* /}</code>	
<code>\readlist\nombres{1+2*3-4/5*6}</code>	
Terme 1 : <code>\nombres[1]\par</code>	Terme 1 : 1
Terme 2 : <code>\nombres[2]</code> (facteurs : <code>\nombres[2,1]</code> et <code>\nombres[2,2]\par</code>	Terme 2 : 2*3 (facteurs : 2 et 3)
Terme 3 : <code>\nombres[3]</code> (facteurs : <code>\nombres[3,1]</code> , <code>\nombres[3,2]</code> et <code>\nombres[3,3]</code>)	Terme 3 : 4/5*6 (facteurs : 4, 5 et 6)

Nombre d'éléments La macro `\listlen<macrolist>[<index>]` nécessite 2 développements pour donner le nombre d'éléments de la liste spécifiée par l'<index>.

La <profondeur> de l'<index> doit être strictement inférieure à celle de la <liste>.

Dans le cas d'un <index> vide, `\listlen<macrolist>[]` donne en 2 développements le même résultat que `<macrolist>len` qui le donne en 1.

<code>\setsepchar{\,/ }</code>	
<code>\readlist\baz{1,2 a b,3 c\4 d e f,5,6\7,,8, ,9 xy z}</code>	a) 3 ou 3
a) <code>\bazlen\</code> ou <code>\listlen\baz[]\par</code>	b) 3
b) <code>\listlen\baz[1]\par</code>	c) 3
c) <code>\listlen\baz[2]\par</code>	d) 5
d) <code>\listlen\baz[3]\par</code>	e) 1
e) <code>\listlen\baz[3,1]\par</code>	f) 2
f) <code>\listlen\baz[3,4]\par</code> 2 éléments vides	g) 3
g) <code>\listlen\baz[3,5]</code>	

Afficher les éléments La macro `\showitems<macrolist>[<index>]` affiche les éléments de la liste spécifiée par <index>, selon le même principe que `\listlen`.

La <profondeur> de l'<index> doit être strictement inférieure à celle de la <liste>.

<code>\setsepchar{\,/ }</code>	a) 1,2 a b,3 c 4 d e f,5,6 7,8, ,9 xy z
<code>\readlist\baz{1,2 a b,3 c\4 d e f,5,6\7,,8, ,9 xy z}</code>	b) 1 2 a b 3 c
a) <code>\showitems\baz[]\par</code>	c) 4 d e f 5 6
b) <code>\showitems\baz[1]\par</code>	d) 7 8 9 xy z
c) <code>\showitems\baz[2]\par</code>	e) 7
d) <code>\showitems\baz[3]\par</code>	f)
e) <code>\showitems\baz[3,1]\par</code>	g) 9 xy z
f) <code>\showitems\baz[3,4]\par</code> 2 éléments vides	
g) <code>\showitems\baz[3,5]</code>	

Éléments vides et espaces extrêmes La suppression des éléments vides et/ou des espaces extrêmes intervient dans *tous* les éléments, quel que soit le degré d'imbrication. Il est clair que choisir un espace comme séparateur est inutile si l'on veut utiliser `\readlist*`. C'est pourquoi dans cet exemple, « * » est choisi comme séparateur.

Dans cet exemple, on ne supprime que les espaces extrêmes en gardant les éléments vides.

<code>\setsepchar{\,/,*}</code>	a) $\overline{1, 2^*a^*b, 3^*c} \overline{4^*d^*e^*f, 5, 6} \overline{7, 8, 9^*xy^*z}$
<code>\readlist* \baz{1, 2*a*b, 3*c \4*d*e*f, 5, 6 \7, , 8, , 9* xy *z}</code>	b) $\overline{1} \overline{2^*a^*b} \overline{3^*c}$
a) <code>\showitems \baz[] \par</code>	c) $\overline{4^*d^*e^*f} \overline{5} \overline{6}$
b) <code>\showitems \baz[1] \par</code>	d) $\overline{7} \overline{8} \overline{9^*xy^*z}$
c) <code>\showitems \baz[2] \par</code>	e) $\overline{7}$
d) <code>\showitems \baz[3] \par</code>	f) $\overline{}$
e) <code>\showitems \baz[3,1] \par</code>	g) $\overline{9} \overline{xy} \overline{z}$
f) <code>\showitems \baz[3,4] \par</code>	
g) <code>\showitems \baz[3,5] % "xy" sans espaces extrêmes</code>	

Itérer sur une liste La syntaxe `\foreachitem <variable> \in <macro>[<index>]{<code>}` reste valable où désormais, l'<index> spécifie sur quel élément (compris comme une liste) on veut itérer.

La <profondeur> de l'<index> doit être strictement inférieure à celle de la <liste>.

Assigner un élément à une macro La syntaxe `\itemtomacro <macroliste>[<index>]<macro>` reste valable pour assigner à <macro> l'élément spécifié par <macroliste>[<index>].

<code>\setsepchar[,]{\\, }</code>	
<code>\readlist \poeme{J'arrive tout couvert encore de rosée\\% Que le vent du matin vient glacer à mon front.\\% Souffrez que ma fatigue à vos pieds reposée\\% Rêve des chers instants qui la délasseront.}% 2e strophe de « Green », Paul Verlaine</code>	2e vers = Que le vent du matin vient glacer à mon front. Un mot = glacer
<code>\itemtomacro \poeme[2] \vers 2e vers = \vers</code>	
<code>\itemtomacro \poeme[2,-4] \mot Un mot = \mot</code>	

La macro `\gitemtomacro` fait une assignation globale.